build
anything
together

Build apps, not platforms: operational maturity in a box

Ori Pekelman

17/11/2022 // Disneyland Paris

platform.sh

// Some assumptions coming into the talk

- By now, for most people coming to a talk like this the cloud is a given: **Cloud Good ✓.**
- By now, for most people coming to a talk like this DevOps practices are a given: **DevOps Good ✓.**

- So we are going to talk about a specific responsibility within Cloud DevOps which is **Platform Engineering ✓**

- **Platform Engineering Good ?**

// Platform Engineering

Whats is it?

“Platform engineering emerged in response to the increasing complexity of modern software architectures. Today, non-expert end users are often asked to operate an assembly of complicated arcane services,”

Says Paul Delory, VP Analyst at Gartner.

“To help end users, and reduce friction for the valuable work they do, forward-thinking companies have begun to build operating platforms that sit between the end user and the backing services on which they rely.“

What defines Operational Maturity?

Its not complete, but let’s try this definition. Operational Maturity is comprised of:

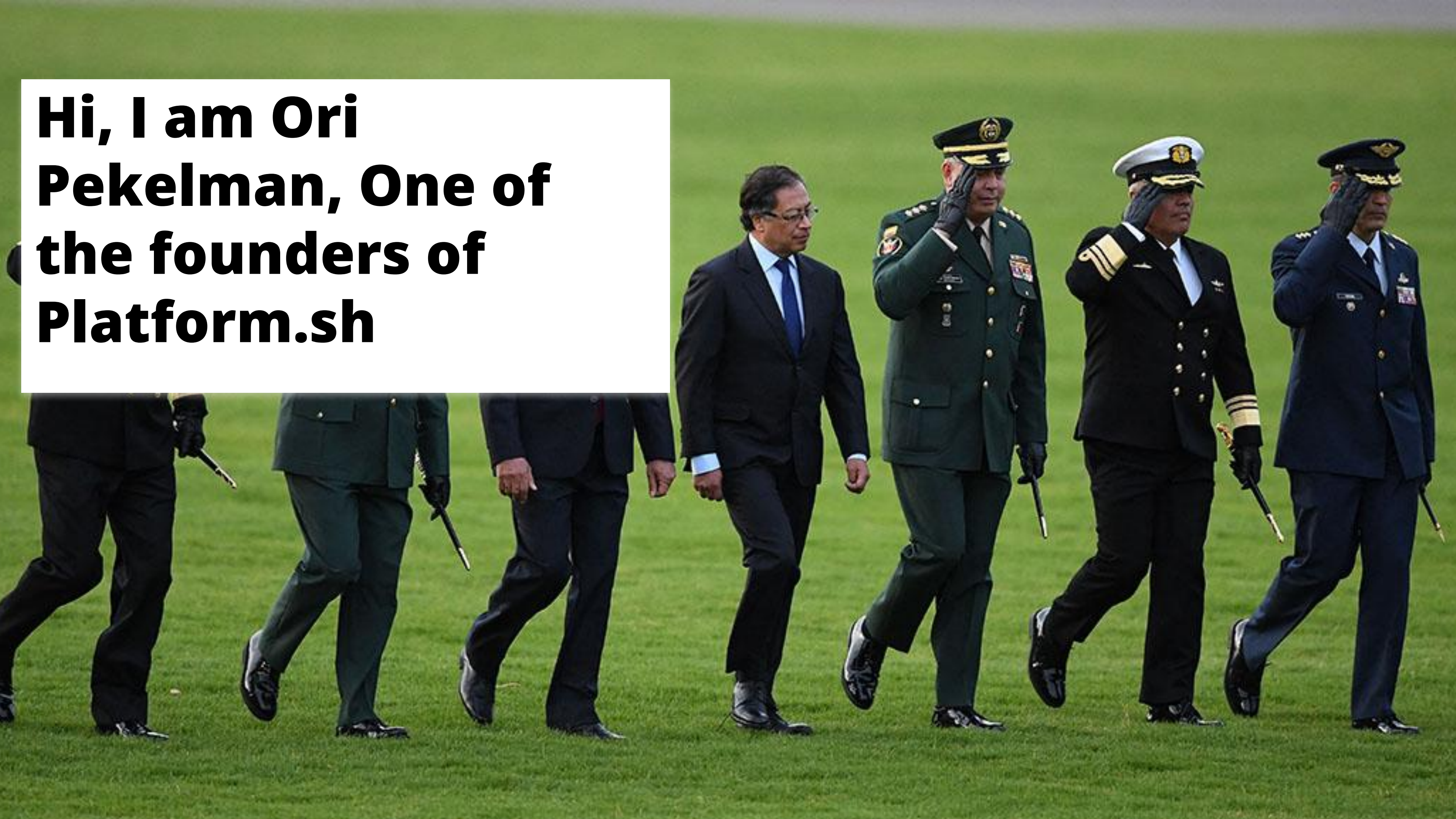
1. A the strict definition of what a non-failing system looks like.
2. Failure Scenarios you test against.
3. Failure modes and their indicators
4. Indicators of the time, cost and end-state of returning to a non-failing state.
5. Handling failures often

How do you achieve Operational Maturity?

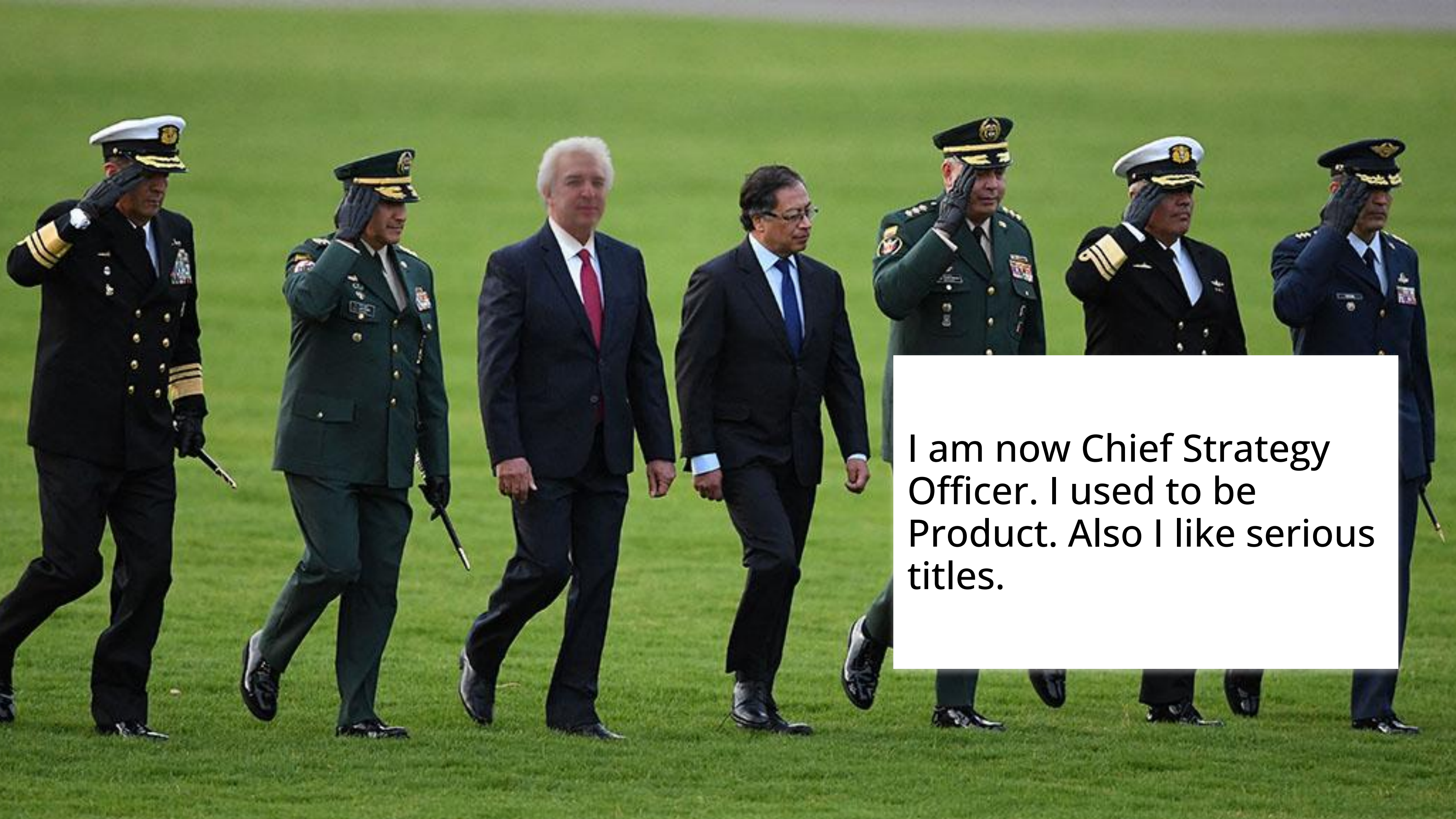
1. Multi-layered observability that tries to capture both positive and negative signals (system is working within defined SLI parameters).
2. Automated failover and mitigations for some of the negative signals
3. People trained to observe these signals and the availability of said people
4. Processes to handle unknown failure modes
5. Run a lot of tests, often - and not only the automated ones

// A segway about me

And a short anecdote



Hi, I am Ori Pekelman, One of the founders of Platform.sh

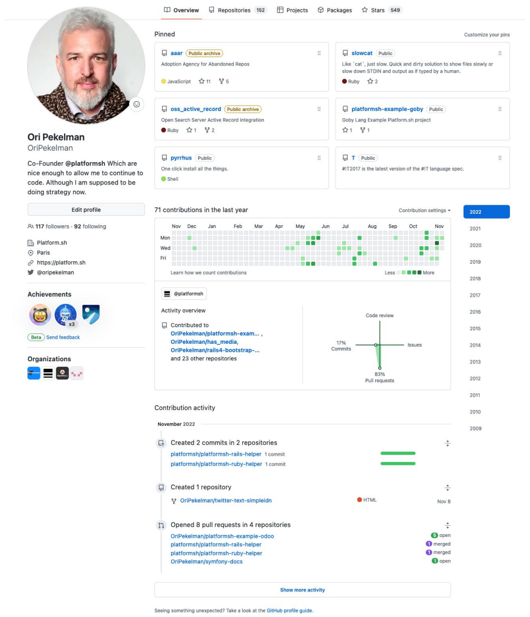


I am now Chief Strategy Officer. I used to be Product. Also I like serious titles.

// **I am also a developer**

* You gotta love stable diffusion

Really.



Overview · Repositories 152 · Projects · Packages · Stars 549

Ori Pekelman
OriPekelman

Co-Founder **@platformsh** Which are nice enough to allow me to continue to code. Although I am supposed to be doing strategy now.

Edit profile

117 followers · 92 following

Platform.sh
Paris
https://platform.sh
@oripekelman

Achievements

Beta Send feedback

Organizations

Pinned — Customize your pins

aaar Public archive
Adoption Agency for Abandoned Repos
JavaScript · ☆ 11 · 5

slowcat Public
Like `cat`, just slow. Quick and dirty solution to show files slowly or slow down STDIN and output as if typed by a human.
Ruby · ☆ 2

oss_active_record Public archive
Open Search Server Active Record integration
Ruby · ☆ 1 · 2

platformsh-example-goby Public
Goby Lang Example Platform.sh project
☆ 1 · 1

pyrrhus Public
One click install all the things.
Shell

T Public
#!T2017 is the latest version of the #!T language spec.

71 contributions in the last year — Contribution settings

Learn how we count contributions — Less · More

2022 2021 2020 2019 2018 2017 2016 2015 2014 2013 2012 2011 2010 2009

@platformsh

Activity overview

Contributed to OriPekelman/platformsh-exam... , OriPekelman/has_media, OriPekelman/rails4-bootstrap-... and 23 other repositories

Contribution activity

November 2022

Created 2 commits in 2 repositories
platformsh/platformsh-rails-helper 1 commit
platformsh/platformsh-ruby-helper 1 commit

Created 1 repository
OriPekelman/twitter-text-simpleidn · HTML · Nov 8

Opened 8 pull requests in 4 repositories
OriPekelman/platformsh-example-odoo · 5 open
platformsh/platformsh-rails-helper · 1 merged
platformsh/platformsh-ruby-helper · 1 merged
OriPekelman/symfony-docs · 1 open

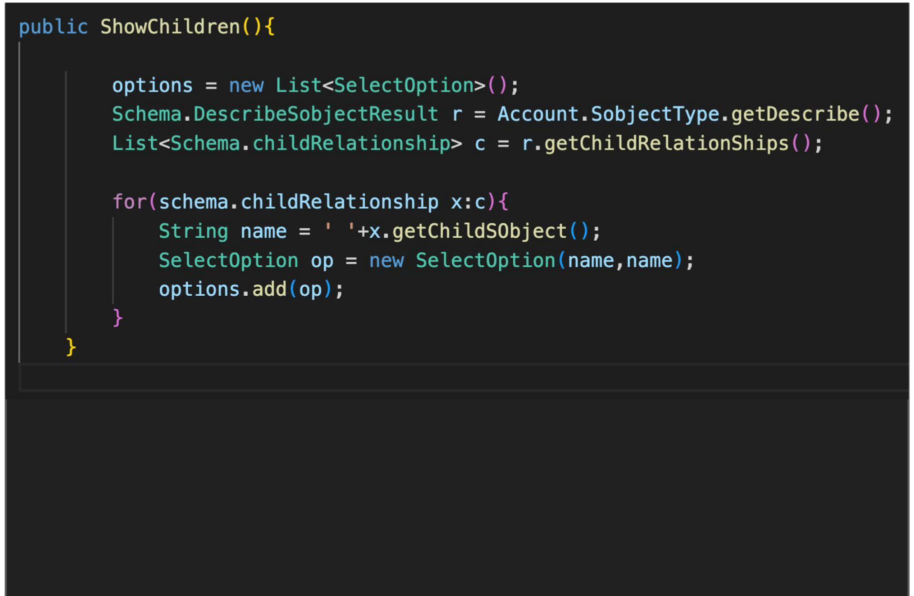
Show more activity

Seeing something unexpected? Take a look at the GitHub profile guide.

// Startups are a wild thing, for a time I ran marketing...

* Apex Predators having a cuteness competition

When you are a developer … and you run marketing…

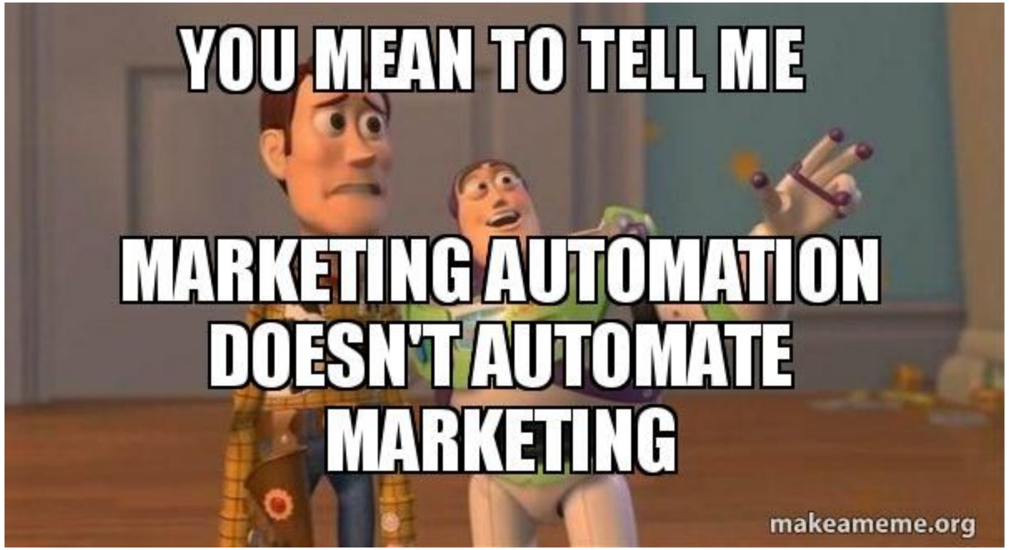


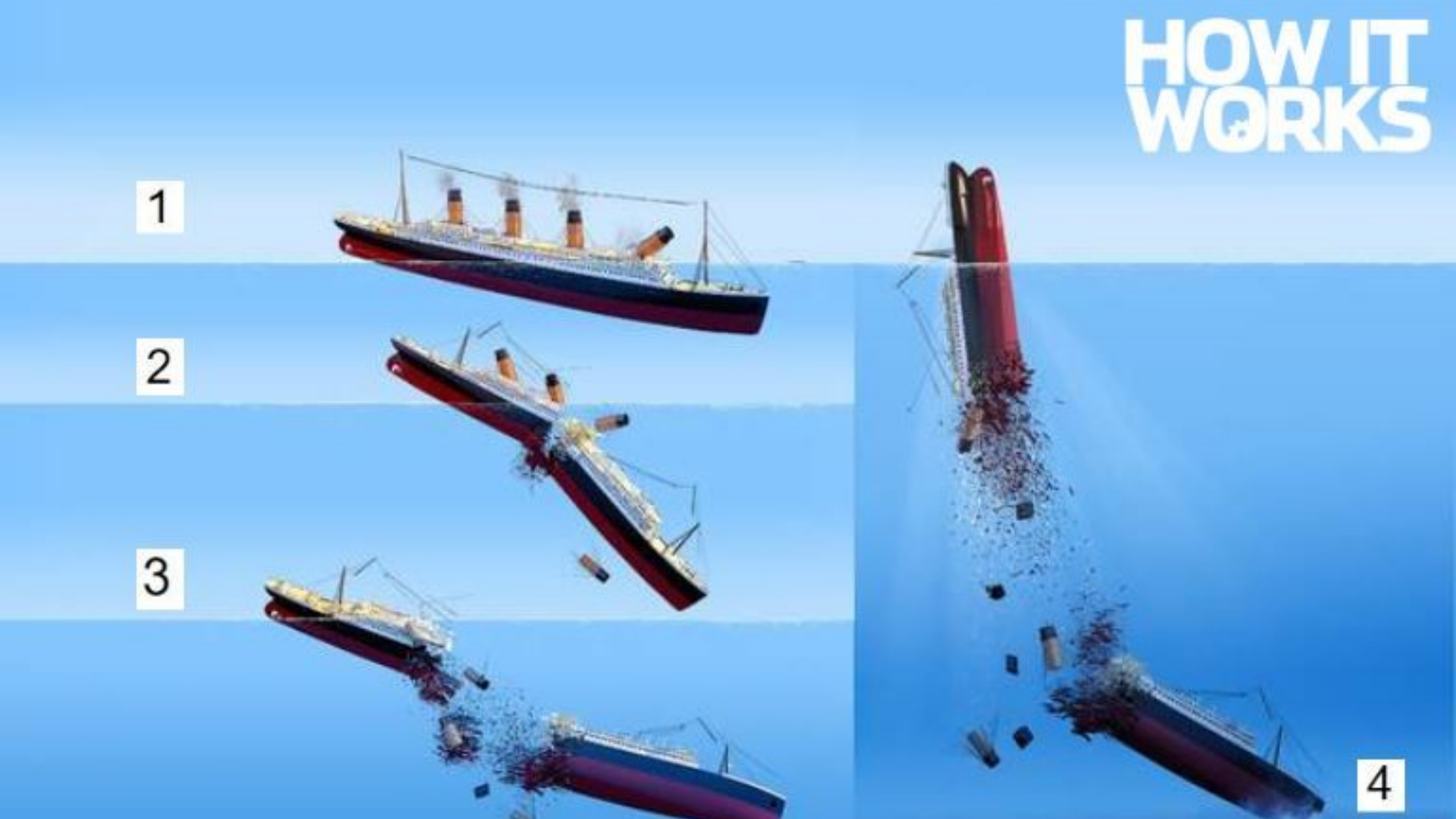
```
public ShowChildren(){

        options = new List<SelectOption>();
        Schema.DescribeSobjectResult r = Account.SobjectType.getDescribe();
        List<Schema.childRelationship> c = r.getChildRelationShips();

        for(schema.childRelationship x:c){
            String name = ' '+x.getChildSObject();
            SelectOption op = new SelectOption(name,name);
            options.add(op);
        }
    }
```

TITANIC

//

When SalesOps is not your day job.




HOW IT WORKS
1
2
3
4

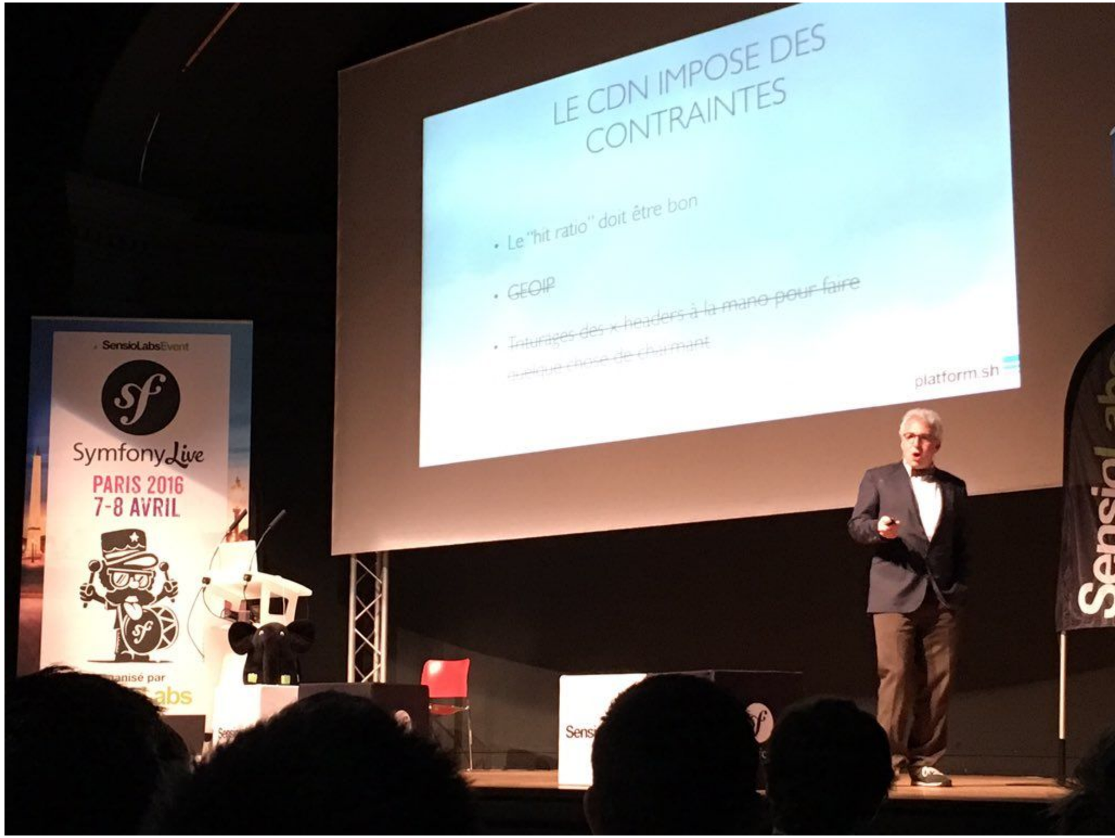
// Back to the actual talk

And to the voyage to Cloud Nativity

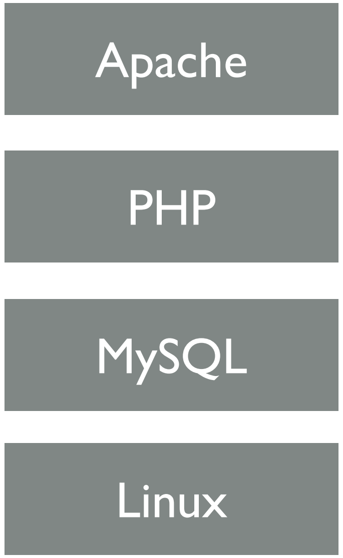
"

Six years ago I did the same presentation here

At the time the title was “How to build cloud native applications” and it was in French but it was basically the same subject.


LE CDN IMPOSE DES CONTRAINTES
Le "hit ratio" doit être bon
~~GEOIP~~
~~Triturages des x-headers à la mano pour faire quelque chose de charmant~~
platform.sh
SensioLabsEvent
SymfonyLive
PARIS 2016
7-8 AVRIL
SensioLabs

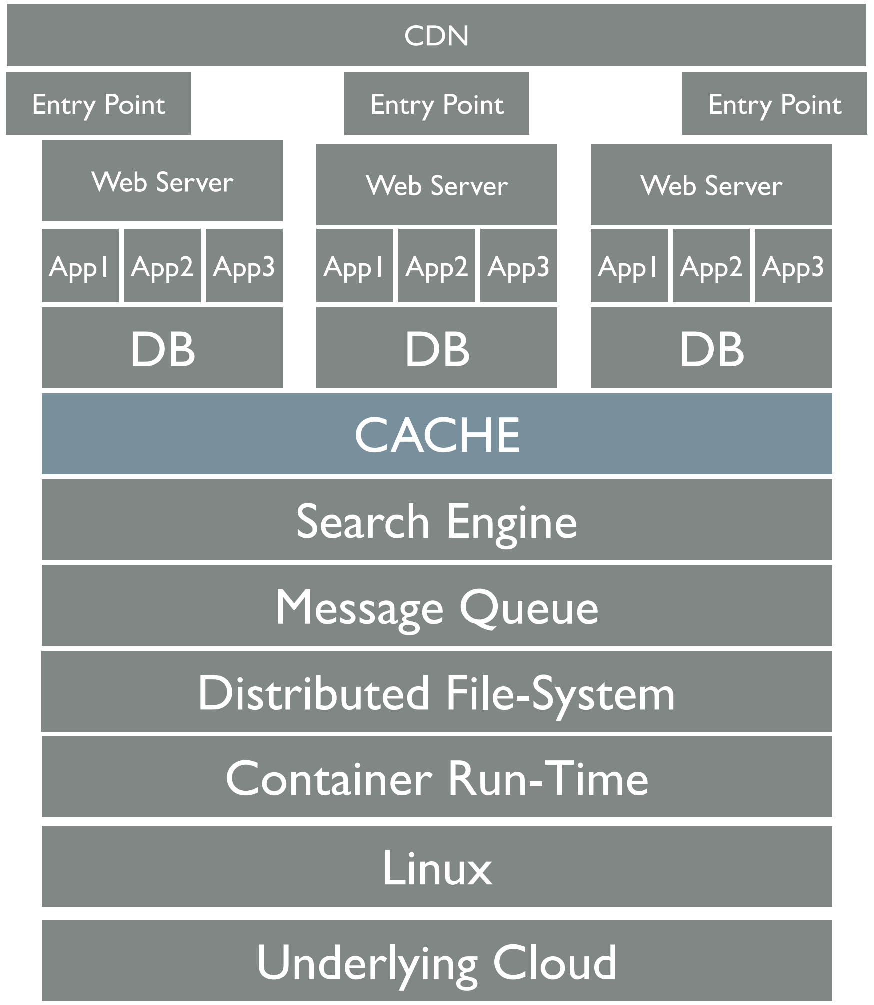
Basically, how and why do we go from...



// To...

And how to keep things simple.

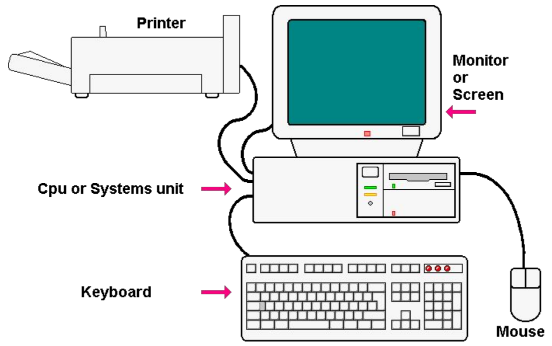
I told things about Infrastructure as Code



Running programs on computers.

// **Computers are simple.**

1. CPU and Memory
2. Disk
3. Networking
4. Processes
5. Names that map to processes exposed on a network.

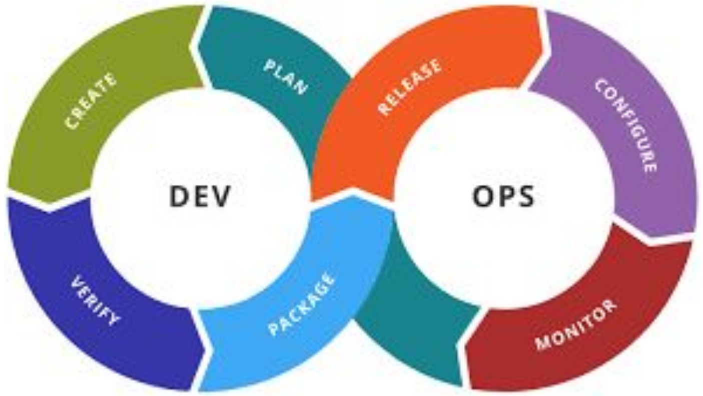


From: David Cushman's The Simplest Possible Tutorial.. Understanding How Computers Work

// **You can add a couple of things to be more complete.**

1. Using CPU memory and disk and exposing themselves for the network that will give them a name
2. Processes are the result of source code
3. Source code needs to be built in order to run

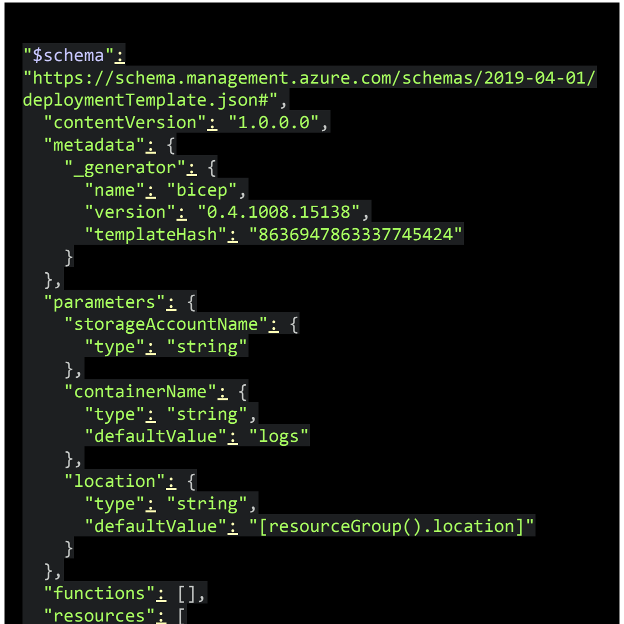
The non-simple part is running programs written by a bunch of humans using third party libraries that change with a certain level of quality within a defined rhythm of change when there are going to be many uses to your program



// **Infrastructure as code to the rescue**

1. We are developers…
2. It’s just code…
3. We can add it under Git control add some tests…
4. And every time the code changes we just run it through a pipe-line…

The Business Domain of Infrastructure as Code



```
"$schema":
"https://schema.management.azure.com/schemas/2019-04-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.4.1008.15138",
      "templateHash": "8636947863337745424"
    }
  },
  "parameters": {
    "storageAccountName": {
      "type": "string"
    },
    "containerName": {
      "type": "string",
      "defaultValue": "logs"
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "functions": [],
  "resources": [
```

1. The same way SalesForce Apex code manages leads and opportunities

2. IaC code manages … well code, and its relationship to the underlying infrastructure

In six years what “cloud native” means changed a lot, the “business domain” evolved.

In 2006 AWS was: S3, EC2, SQS

By 2009 SimpleDB, Elastic IPs, EBS, Cloud Front, Amazon Elastic MapReduce, ELB, VPCs, RDS

By 2012 SNS, CloudFormation, Route 53, Elastic Beanstalk, SES, DynamoDB, IAM, Glacier, RedShift

By 2015 CloudHSM, Cloud Trail, Kinesis, Aurora, KMS, ECS, Lambda, CodePipeline, API Gateway, Elastic Search, Inspector, Snowball, ECR, ML Platform, IOT Platform, Auto Scaling, ACM, EFS....

In 2022 there are more than

289 **distinct**

products.

Just from AWS.

Six years ago

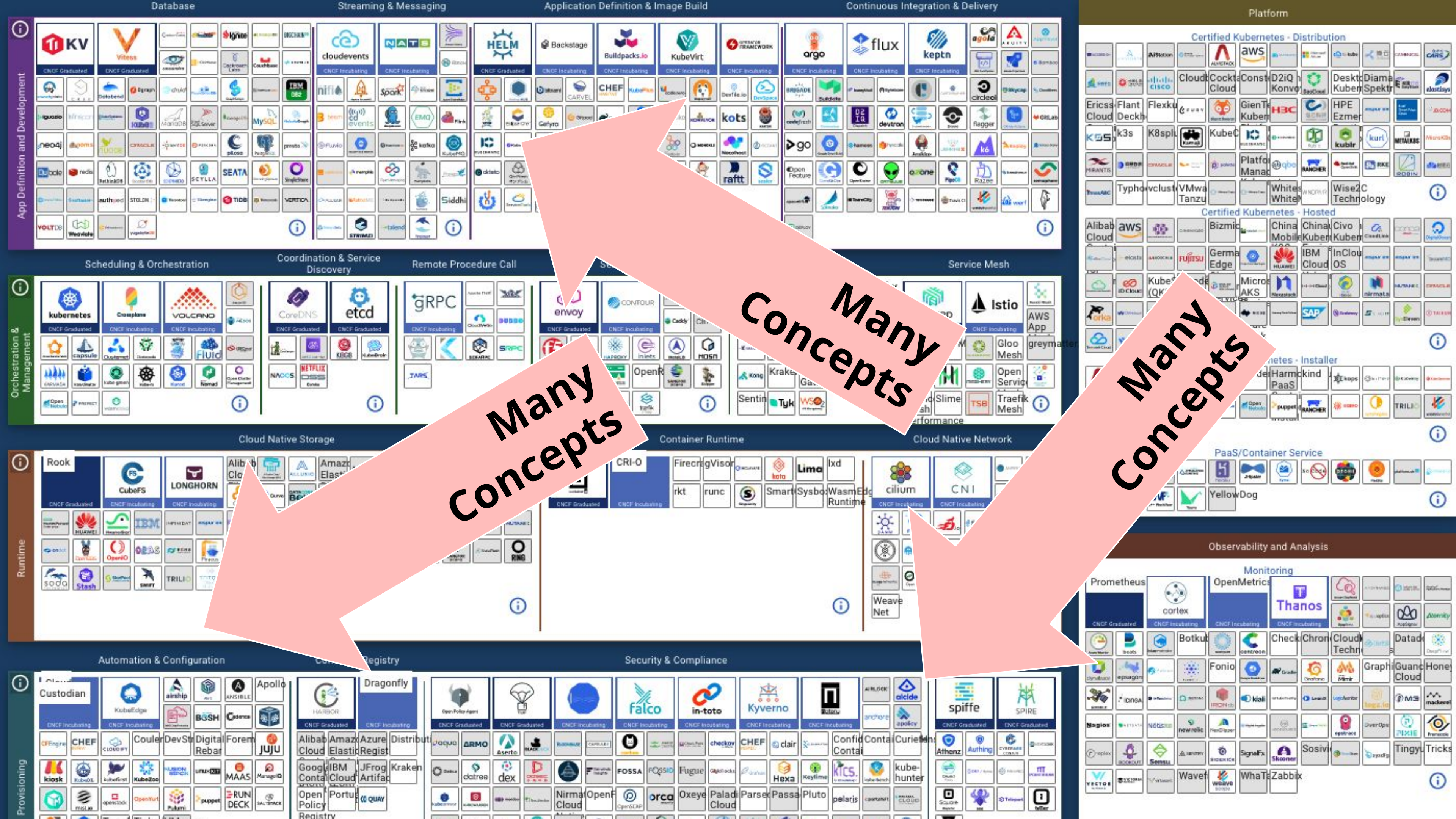
I contrasted LAMP to what a “modern architecture” would look like.

~ 4 Concepts

Adding into the mix an Edge Layer, Multi-tiered Caching, a message queue, Redundant Storage and Replicated databases, and a Converged Storage Layer.

+8 Concepts

All with an integrated CI/CD, backed by GitOps


Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Platform
App Definition and Development
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Mesh
Orchestration & Management
Cloud Native Storage
Container Runtime
Cloud Native Network
Runtime
PaaS/Container Service
Observability and Analysis
Monitoring
Automation & Configuration
Registry
Security & Compliance
Provisioning
Many Concepts
Many Concepts
Many Concepts

Platform Engineering

Whats is it?

Gartner expects that by 2026, **80%** of software engineering organizations will establish platform teams as internal providers of reusable services, components and tools for application delivery.

Platform engineering will ultimately **solve** the central problem of cooperation between software developers and operators.

Platforms Vs DevOps

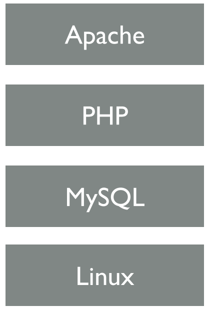
What is the difference between DevOps and Platform Engineering?

Contrast this to what Gartner said about DevOps in 2016:

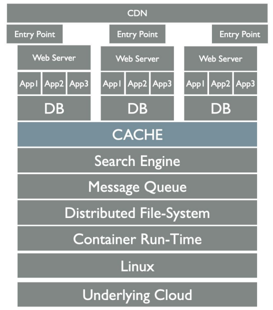
Organizations with agile development will be slower to embrace DevOps across the entire application life cycle. Cultural resistance and low levels of process discipline will create significant failure rates for DevOps initiatives, particularly when waterfall processes are still a dominant portion of the development portfolio. Nevertheless, a majority of enterprises attempting to scale agile over the next five years will recognize the need for DevOps initiatives.

// What “Platforms” mean also changed.

Platforms in 2008 needed to basically handle:

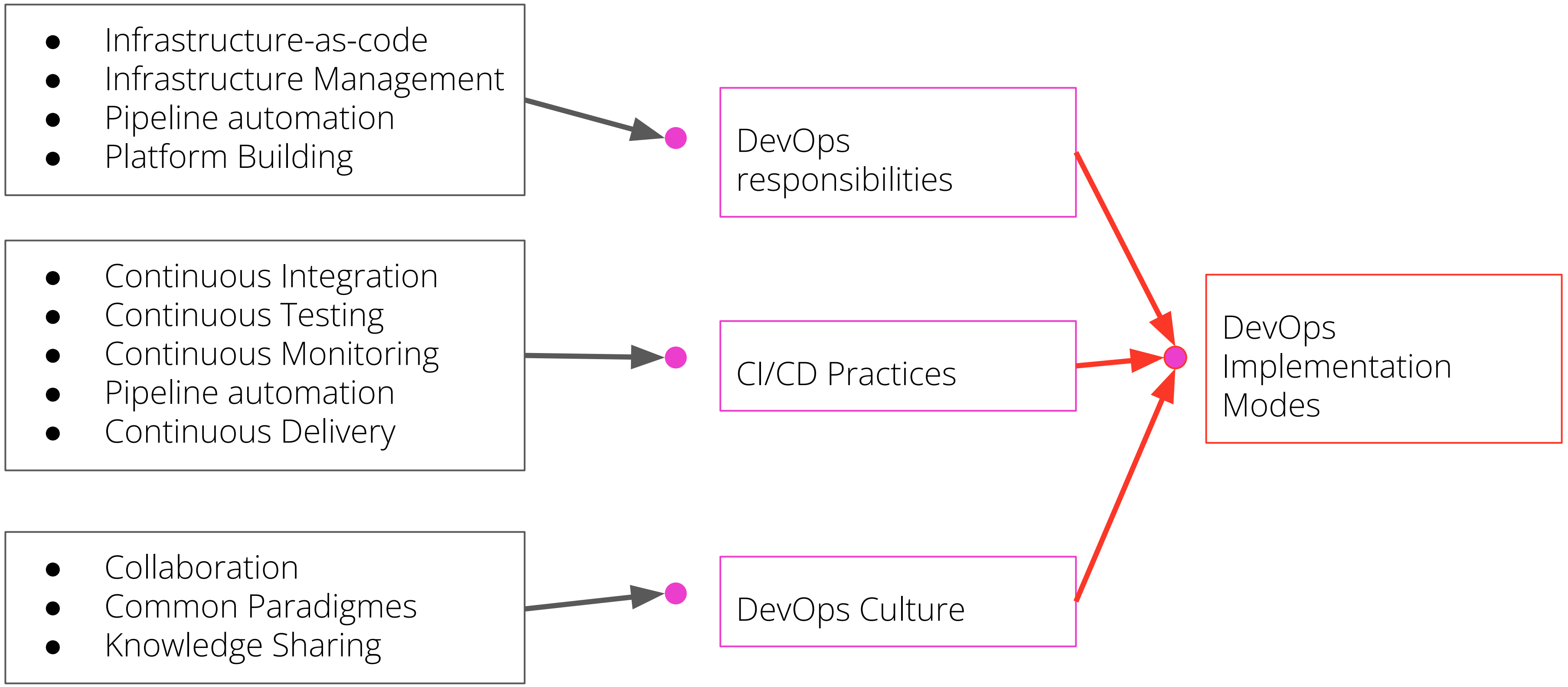


In 2016 they needed to do this, as a bare minimum:



And that is before you run any machine-learningy things at the edge. Before you get into consideration dynamic scaling and handling new forms of DDOS attacks. Before you consider your carbon footprint. Before you consider cost.

DevOps As a Job Desc, DevOps as a culture



DevOps vs Platform Engineering

What is the difference between DevOps and Platform Engineering?

DevOps is a philosophy, a cultural shift that merges operations with development and demands a linked toolchain of technologies to facilitate collaborative change.

Platform Engineering is solving all of the huge problems that arose as soon as the above became “use Kubernetes” and a dozen or so tools to “simplify Kubernetes”.

But it’s also back to 2008 and the promises the cloud initially had about simplicity.

Roles and responsibilities

	Developers	SREs	DevOps	Ops	IT
Code	.	.	.	.	
Continuous Integration	.	.	.	.	.
Deployment	.	.	.	.	.
Incident Management	.	.	.	.	.
Performance Management	.	.	.	.	.
Infrastructure Management	.	.	.	.	.
Cost Management	.	.	.	.	.

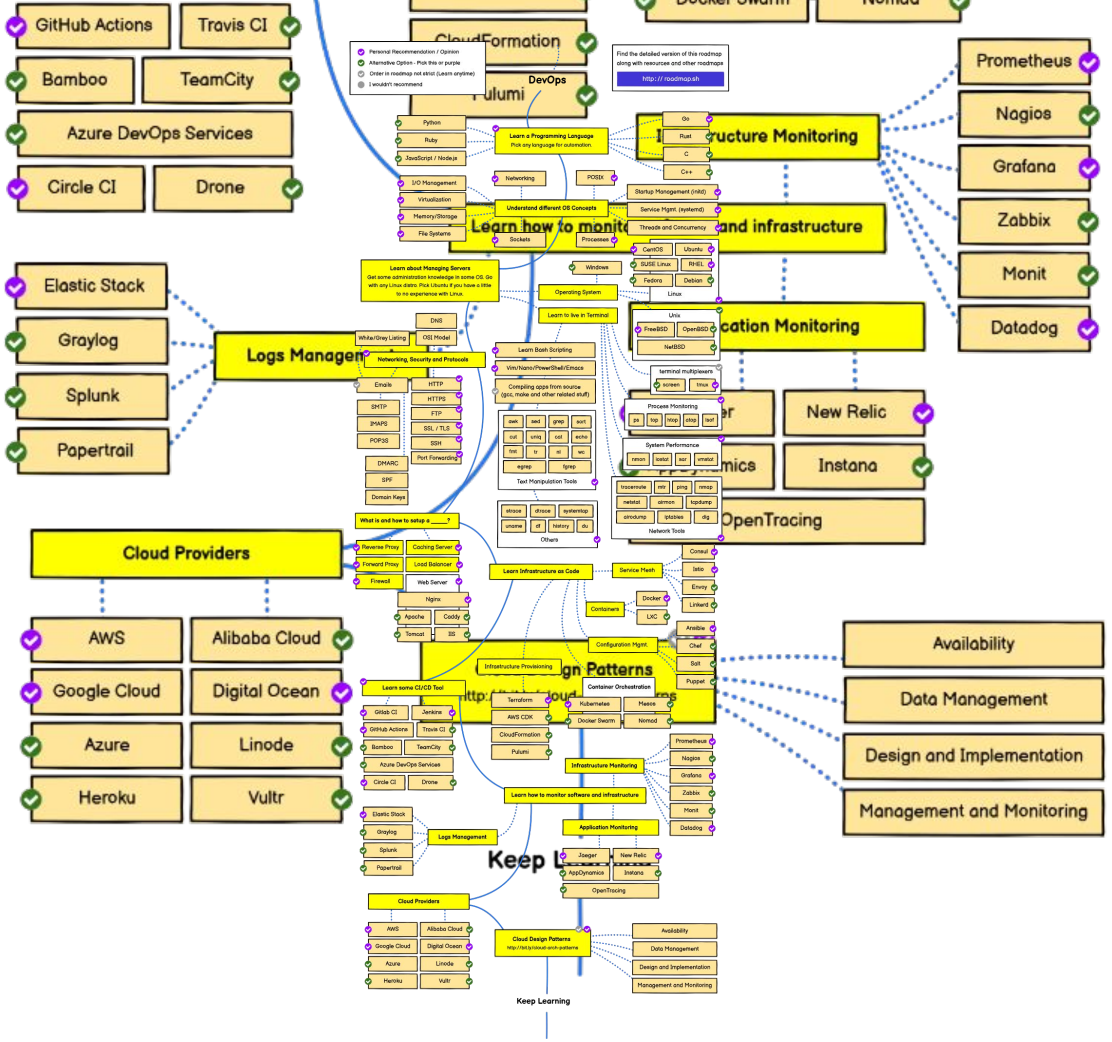
"

In six years what "cloud native" means changed a lot

Developers

Well, not that much changed for us. We still just write code. And tests. In whatever order.

"Our job stayed the same. But I can't say their's became any easier



// **In six years what "cloud native" means changed a lot**

"On the other side" even the titles of the people actually making it happen changed quite a bit ...

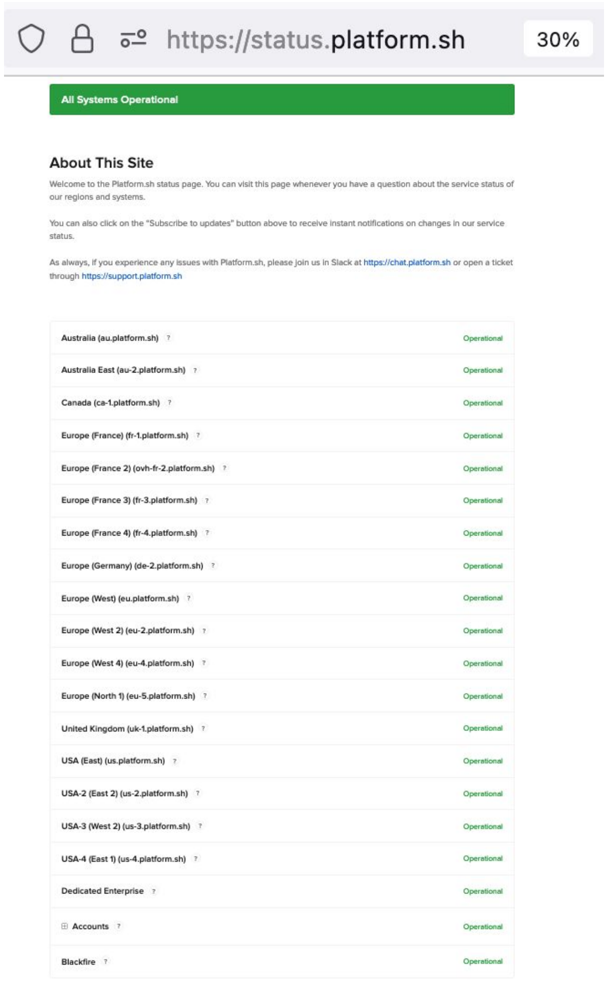
Platform Engineer

// Operational maturity is not about green

And it's not about not having failures.

Everything fails. All the time. This is cloud and this is software.

Everything is horrendously broken.



https://status.platform.sh 30%

All Systems Operational

About This Site

Welcome to the Platform.sh status page. You can visit this page whenever you have a question about the service status of our regions and systems.

You can also click on the "Subscribe to updates" button above to receive instant notifications on changes in our service status.

As always, if you experience any issues with Platform.sh, please join us in Slack at https://chat.platform.sh or open a ticket through https://support.platform.sh

Australia (au.platform.sh) ?	Operational
Australia East (au-2.platform.sh) ?	Operational
Canada (ca-1.platform.sh) ?	Operational
Europe (France) (fr-1.platform.sh) ?	Operational
Europe (France 2) (ovh-fr-2.platform.sh) ?	Operational
Europe (France 3) (fr-3.platform.sh) ?	Operational
Europe (France 4) (fr-4.platform.sh) ?	Operational
Europe (Germany) (de-2.platform.sh) ?	Operational
Europe (West) (eu.platform.sh) ?	Operational
Europe (West 2) (eu-2.platform.sh) ?	Operational
Europe (West 4) (eu-4.platform.sh) ?	Operational
Europe (North 1) (eu-5.platform.sh) ?	Operational
United Kingdom (uk-1.platform.sh) ?	Operational
USA (East) (us.platform.sh) ?	Operational
USA-2 (East 2) (us-2.platform.sh) ?	Operational
USA-3 (West 2) (us-3.platform.sh) ?	Operational
USA-4 (East 1) (us-4.platform.sh) ?	Operational
Dedicated Enterprise ?	Operational
⊞ Accounts ?	Operational
Blackfire ?	Operational

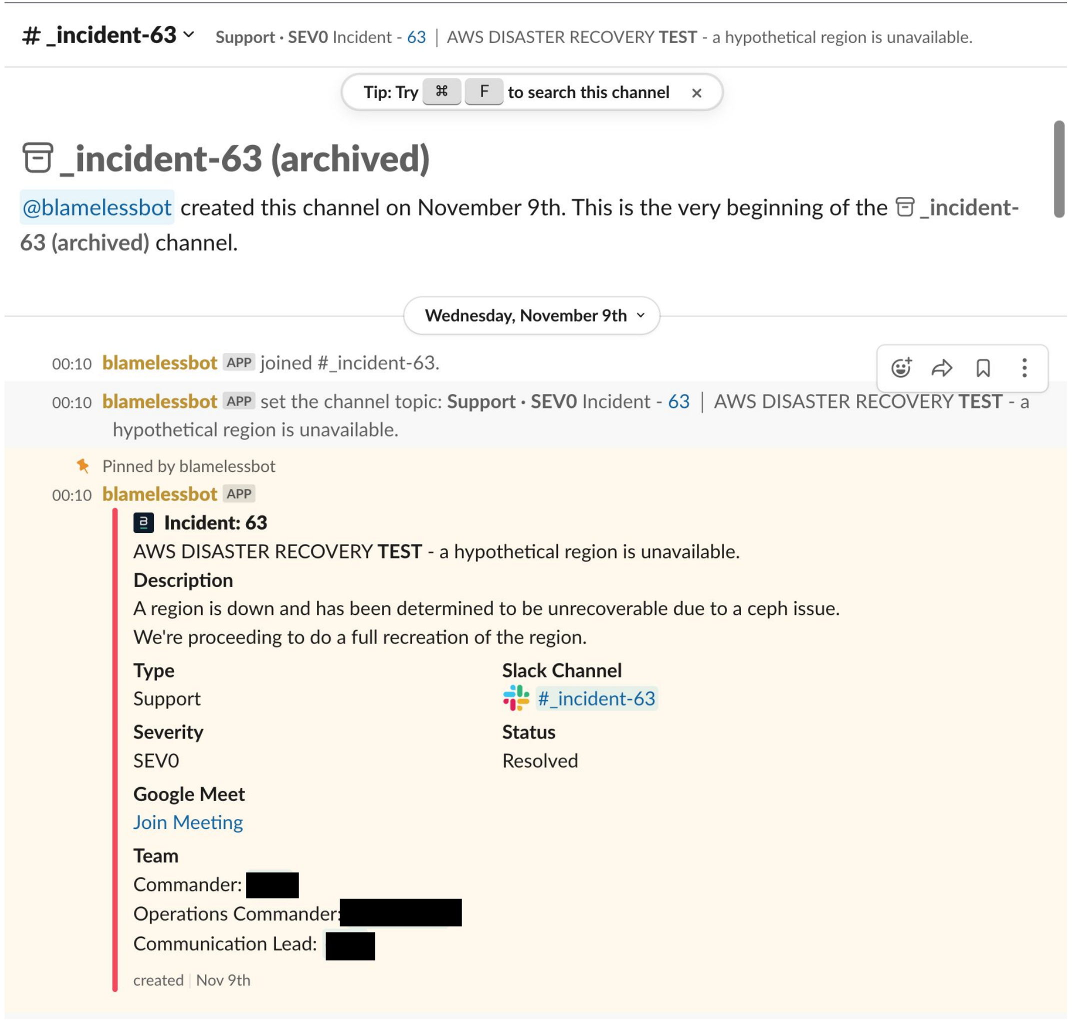
// It's about red

Having a lot of automation for the normal kind of red.

Disks frying. Hosts dying.

And having people that can handle a new shade of red.

You know, unknown unknowns.



_incident-63 Support · **SEV0** Incident - 63 | AWS DISASTER RECOVERY **TEST** - a hypothetical region is unavailable.

Tip: Try ⌘ F to search this channel ×

_incident-63 (archived)

@blamelessbot created this channel on November 9th. This is the very beginning of the _incident-63 (archived) channel.

Wednesday, November 9th

00:10 **blamelessbot** APP joined #_incident-63.

00:10 **blamelessbot** APP set the channel topic: **Support · SEV0** Incident - 63 | AWS DISASTER RECOVERY **TEST** - a hypothetical region is unavailable.

Pinned by blamelessbot

00:10 **blamelessbot** APP

Incident: 63

AWS DISASTER RECOVERY **TEST** - a hypothetical region is unavailable.

Description

A region is down and has been determined to be unrecoverable due to a ceph issue. We're proceeding to do a full recreation of the region.

Type	**Slack Channel**
Support	#_incident-63
Severity	**Status**
SEV0	Resolved

Google Meet

Join Meeting

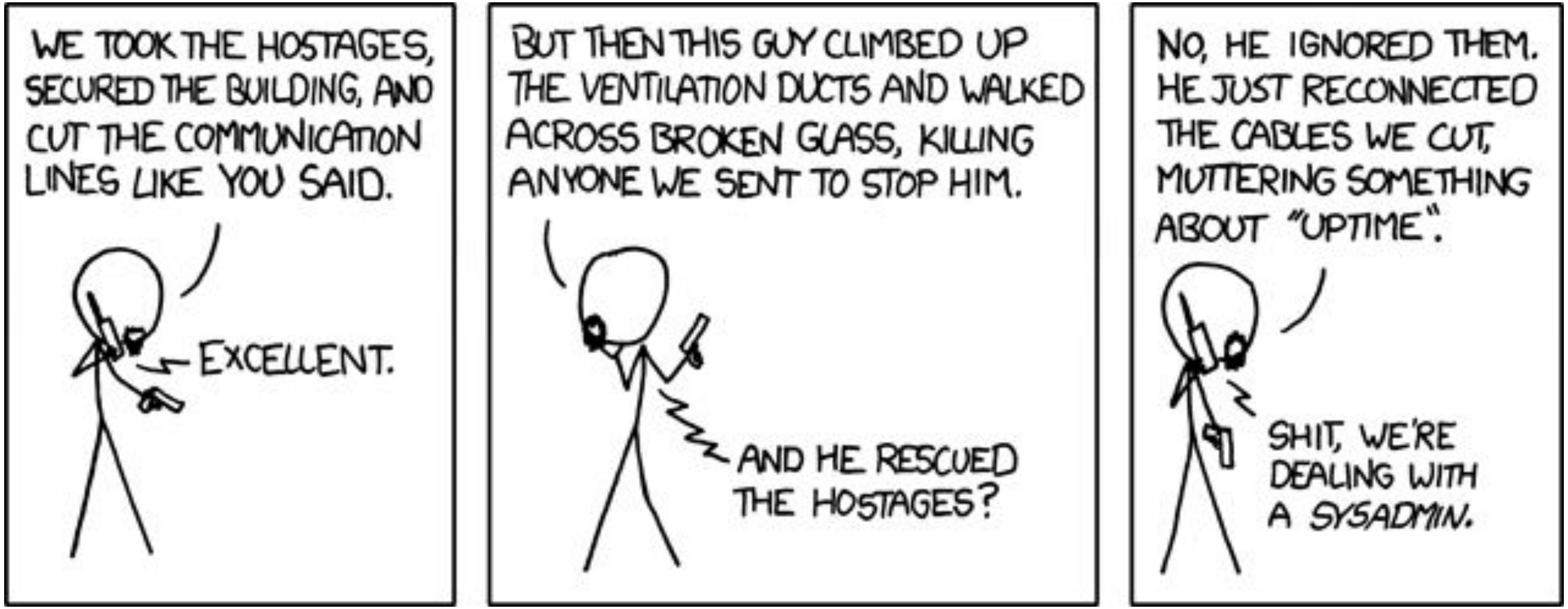
Team

Commander: [redacted]

Operations Commander: [redacted]

Communication Lead: [redacted]

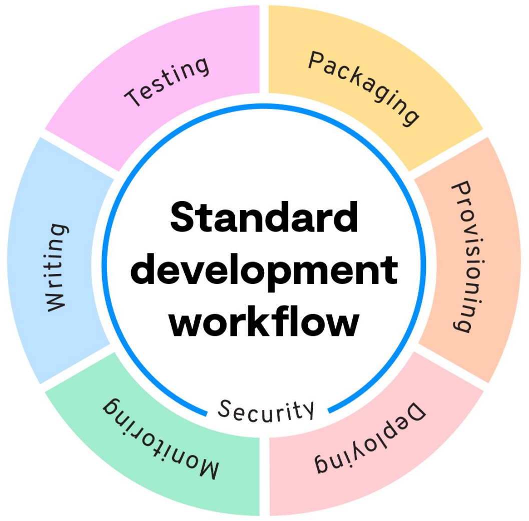
created | Nov 9th


WE TOOK THE HOSTAGES, SECURED THE BUILDING, AND CUT THE COMMUNICATION LINES LIKE YOU SAID.
EXCELLENT.
BUT THEN THIS GUY CLIMBED UP THE VENTILATION DUCTS AND WALKED ACROSS BROKEN GLASS, KILLING ANYONE WE SENT TO STOP HIM.
AND HE RESCUED THE HOSTAGES?
NO, HE IGNORED THEM. HE JUST RECONNECTED THE CABLES WE CUT, MUTTERING SOMETHING ABOUT "UPTIME".
SHIT, WE'RE DEALING WITH A *SYSADMIN*.

// Ah, is this the actual talk yet?

Yes, almost ....

// Standard deployment workflow



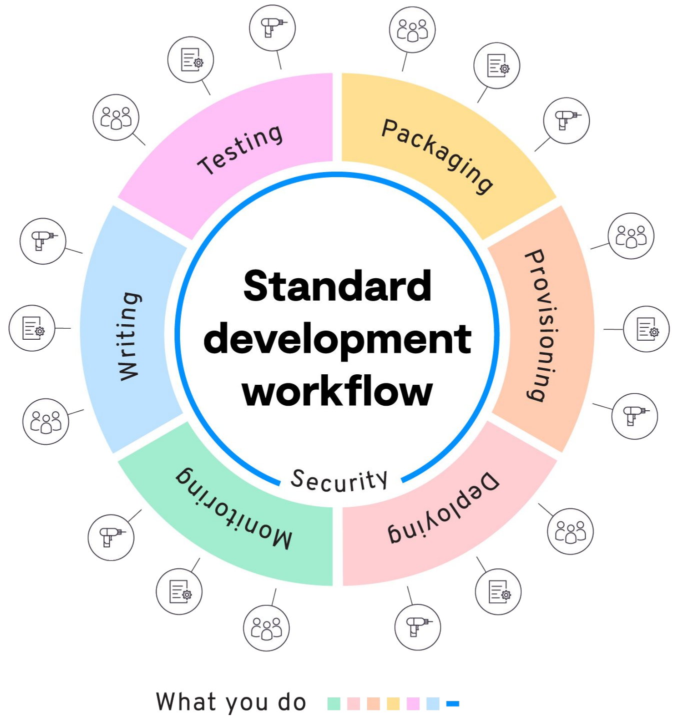
What you do 🟩 🟥 🟧 🟨 🟪 🟦 ▬

Standard deployment workflow

// Building in-house

Each phase requires people, configuration and tooling to make it run efficiently and consistently.

And now you need to do it twice.



// Not all code is equal and semantics matter

```
# .platform.app.yaml
name: ‘symfony’
type: ‘php:8.1’
relationships:
  database: ‘db:postgresql’
mounts:
   "/var/cache": "shared:files/local”
# .platform/services.yaml
db:
    type: postgresql:13
    disk: 2048
```

// And it’s not just about verbosity

When the infrastructure is a dependency the contract matters.

Are **PORTS** part of my software definition? Part of the infrastructure definition?

Is it Apache that is my dependency or a reverse HTTP ?Am I locked to minor versions? To major ones?

```
version: "3.8"
services:
   db:
       image: mysql
       container_name: db_docker_symfony
       restart: always
       volumes:
           - db-data:/var/lib/mysql
       environment:
           MYSQL_ALLOW_EMPTY_PASSWORD: 'yes'
       networks:
           - dev
   www:
       build: php
       container_name: www_docker_symfony
       ports:
         - "8741:80"
       volumes:
           - ./php/vhosts:/etc/apache2/sites-enabled
           - ./:/var/www
       restart: always
       networks:
           - dev
networks:
   dev:
volumes:
   db-data:
```

How reproducible are you?

The style of code, its semantics are going to have a huge impact down the line.

Descriptive and imperative styles are not the same.

And their relationship to version control is paramount.

```yaml
- name: Setting up LAMP Website
  user: symfony
  hosts: testserver
  become: yes
  tasks:
    - name: latest version of all required packages installed
      yum:
        name:
          - httpd
          - mariadb-server
          - php
          - php-mysql
        state: latest

    - name: Copy mime.types file
      copy:
        src: /etc/mime.types
        dest: /etc/httpd/conf/mime.types
        remote_src: yes

    - name: httpd enabled and running
      service:
        name: httpd
        enabled: true
        state: started

    - name: mariadb enabled and running
      service:
        name: mariadb
        enabled: true
        state: started

    - name: test the webpage/website we have setup
      uri:
        url: http://{{ansible_hostname}}/index.php
        status_code: 200
```

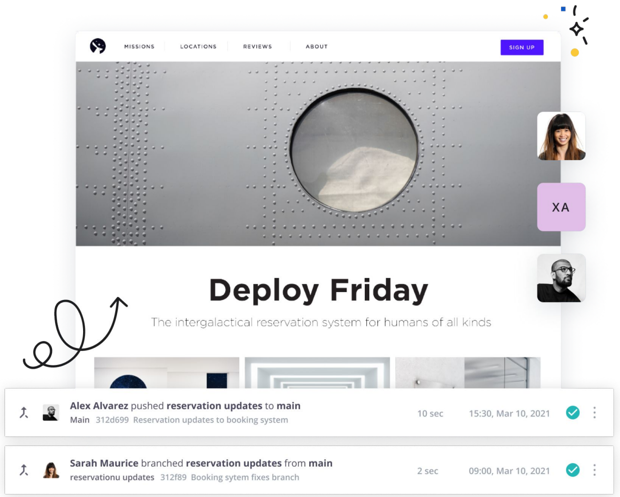
// What happens when we change …

```
# .platform.app.yaml
name: ‘symfony’
type: ‘php:8.1’
relationships:
  database: ‘db:postgresql’
mounts:
   "/var/cache": "shared:files/local”
# .platform/services.yaml
db:
    type: postgresql:13
    disk: 2048
```

// To...

```
# .platform.app.yaml
name: ‘symfony’
type: ‘php:8.2’
relationships:
  database: ‘db:postgresql’
mounts:
   "/var/cache": "shared:files/local”
# .platform/services.yaml
db:
    type: postgresql:14
    disk: 2048
```

What is Platform.sh?


MISSIONS
LOCATIONS
REVIEWS
ABOUT
SIGN UP
XA
Deploy Friday
The intergalactical reservation system for humans of all kinds

Alex Alvarez pushed reservation updates to main
Main 312d699 Reservation updates to booking system 10 sec 15:30, Mar 10, 2021

Sarah Maurice branched reservation updates from main
reservationu updates 312f89 Booking sytem fixes branch 2 sec 09:00, Mar 10, 2021

Platform.sh delivers a framework (Platform-as-a-Service) to **build, run, and effortlessly scale web applications.**

Platform.sh is a multi-cloud **software orchestration solution** that encapsulates the **full life-cycle** of a software project. Including all of the dependencies, from the first lines of code to run & scale.

It targets the specific use-case of organizations that manage a large number of web applications and web sites.
It is an **abstraction** of everything software needs in order to run.
It is a **contract** that explains how a particular piece of software can be run. It is a **control plane** and a single pane of glass.

Its ambition is to help developers develop, deploy and manage with ease not only singular projects but also **fleets of applications**.

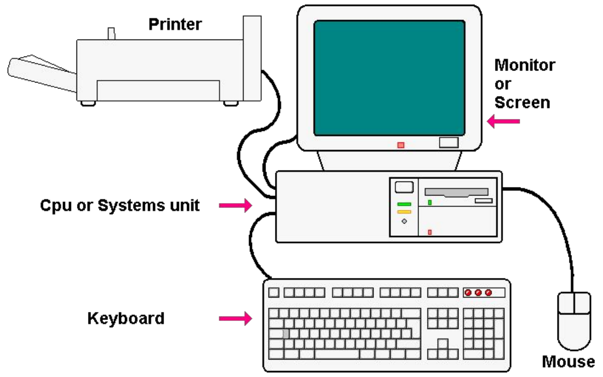
You still want to build a platform?

A note about APIs, system boundaries and the double control plane.

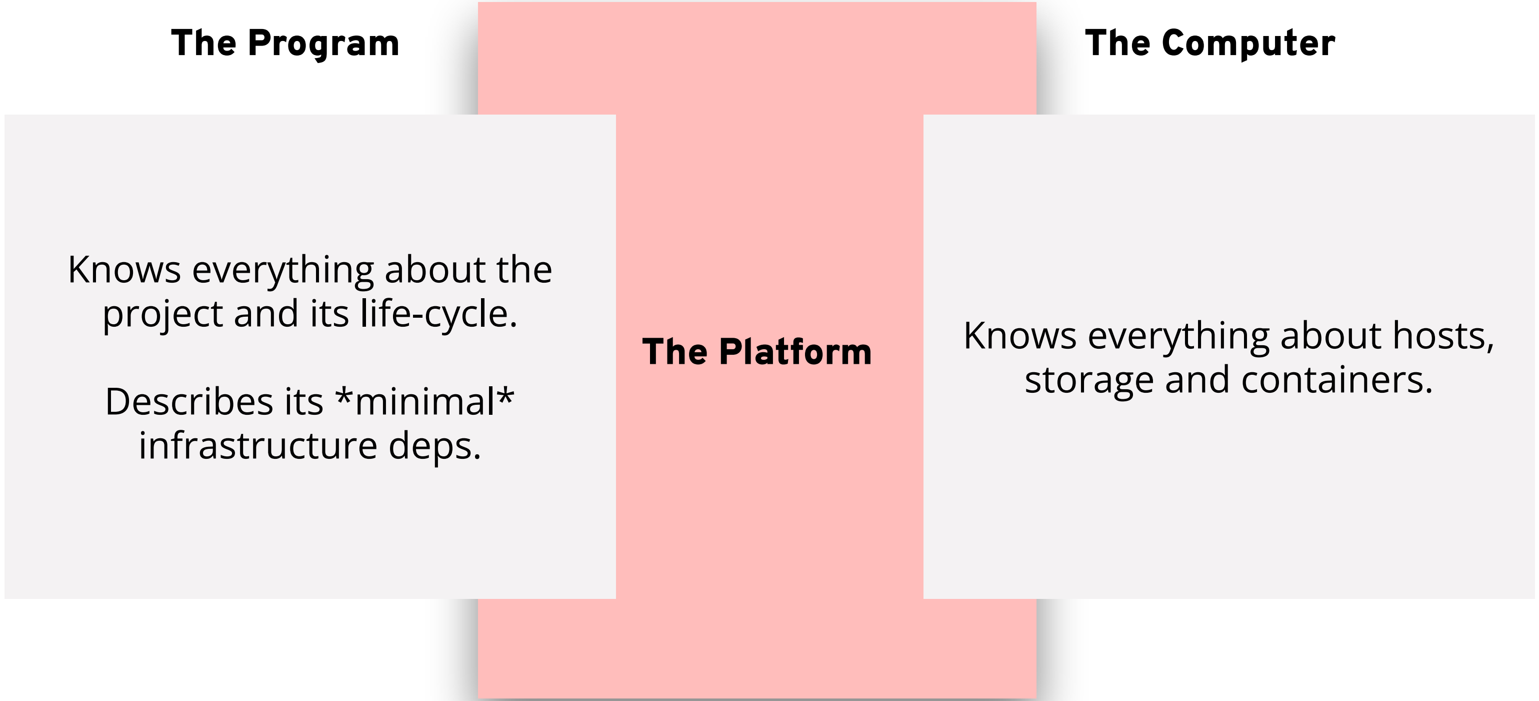
Project Control Plane

Infrastructure Control Plane

Remember this?



Correct system boundaries are key.



The Program

Knows everything about the project and its life-cycle.

Describes its *minimal* infrastructure deps.

The Platform

The Computer

Knows everything about hosts, storage and containers.

To the growing complexity, the answer has been: more tools and greater complexity.

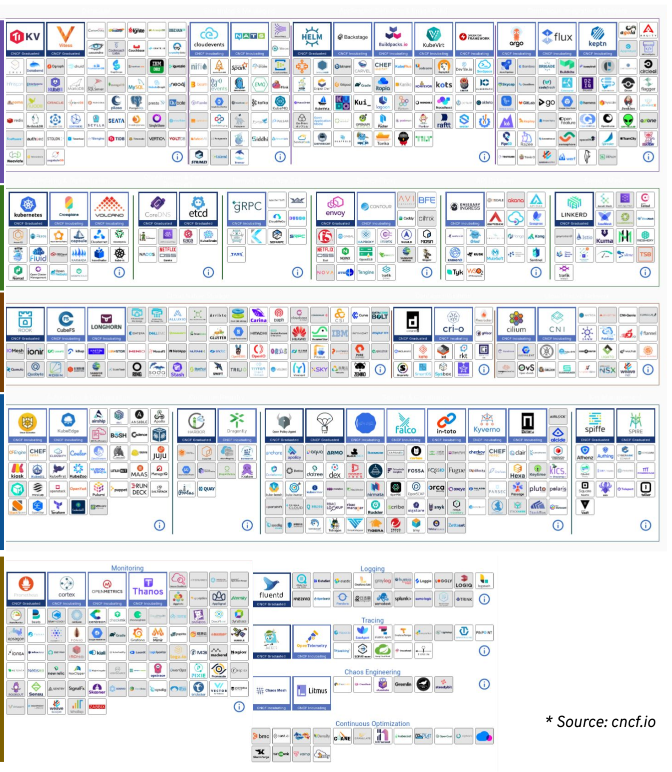
App Definition and Development

Orchestration and management

Runtime

Provisioning

Observability



* *Source: cncf.io*

A disconnected patchwork of tools.

Each with its own learning curve.

Each with its own quirks, SLAs, pricing schemes and lockin risks.

In a regulatory environment that is becoming ever increasingly stringent.

PCI DSS

AICPA SOC

GDPR

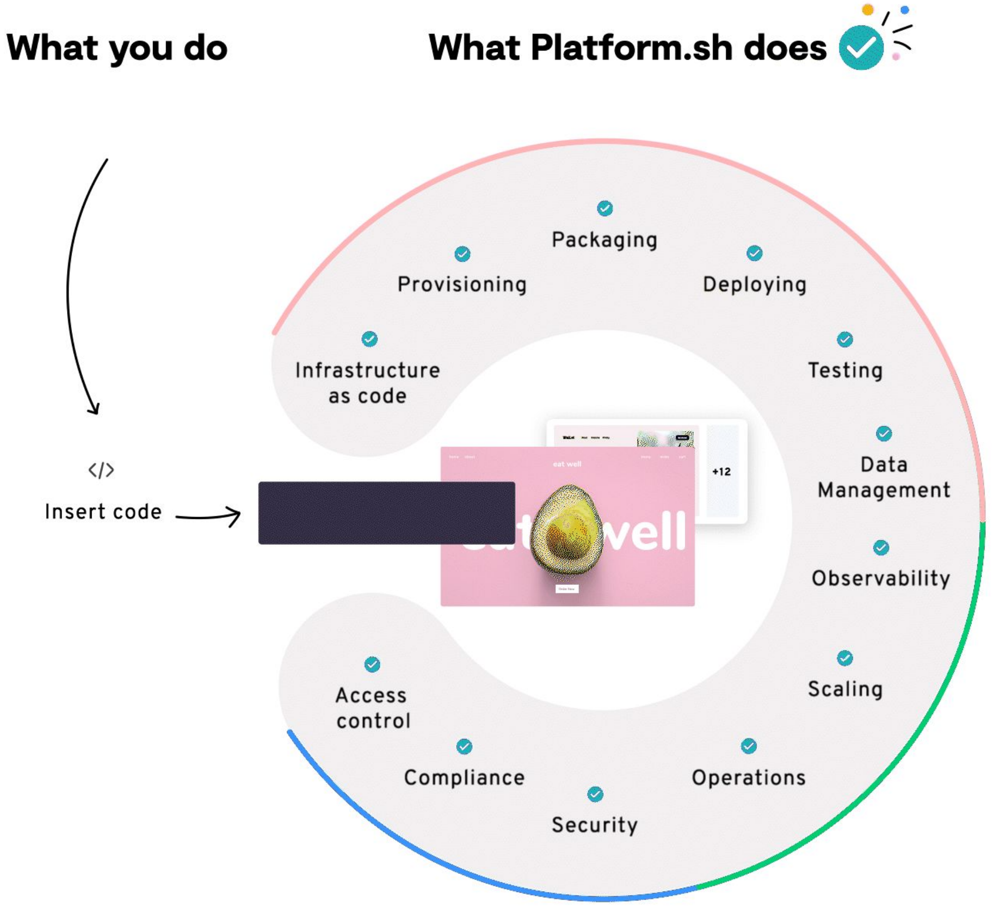
PIPEDA

HIPAA

On average, DevOps teams use between 10 and 15 tools

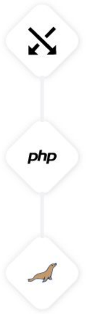
Developers bring their code, we bring the rest

Platform.sh offers a unified, secure, enterprise-grade platform for responsibly building, running and scaling fleets of websites and applications.

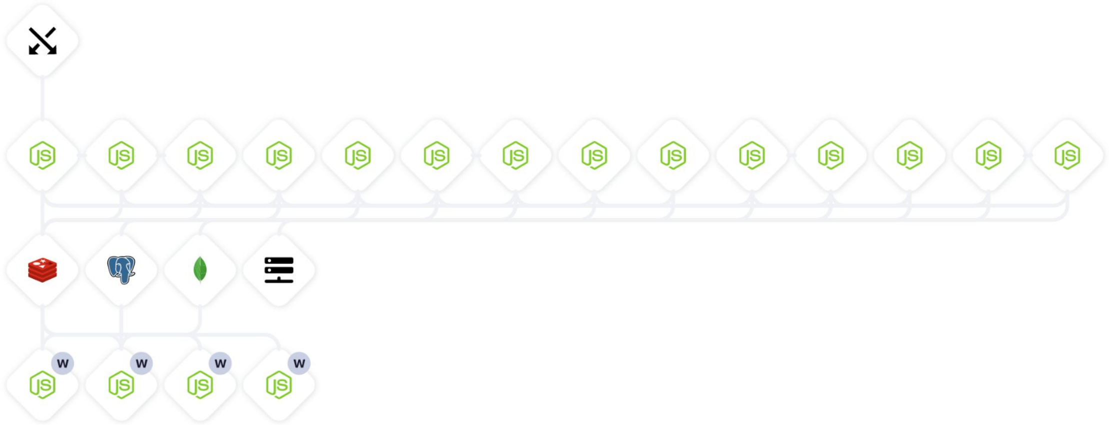


From Monoliths to anything..

Just an App

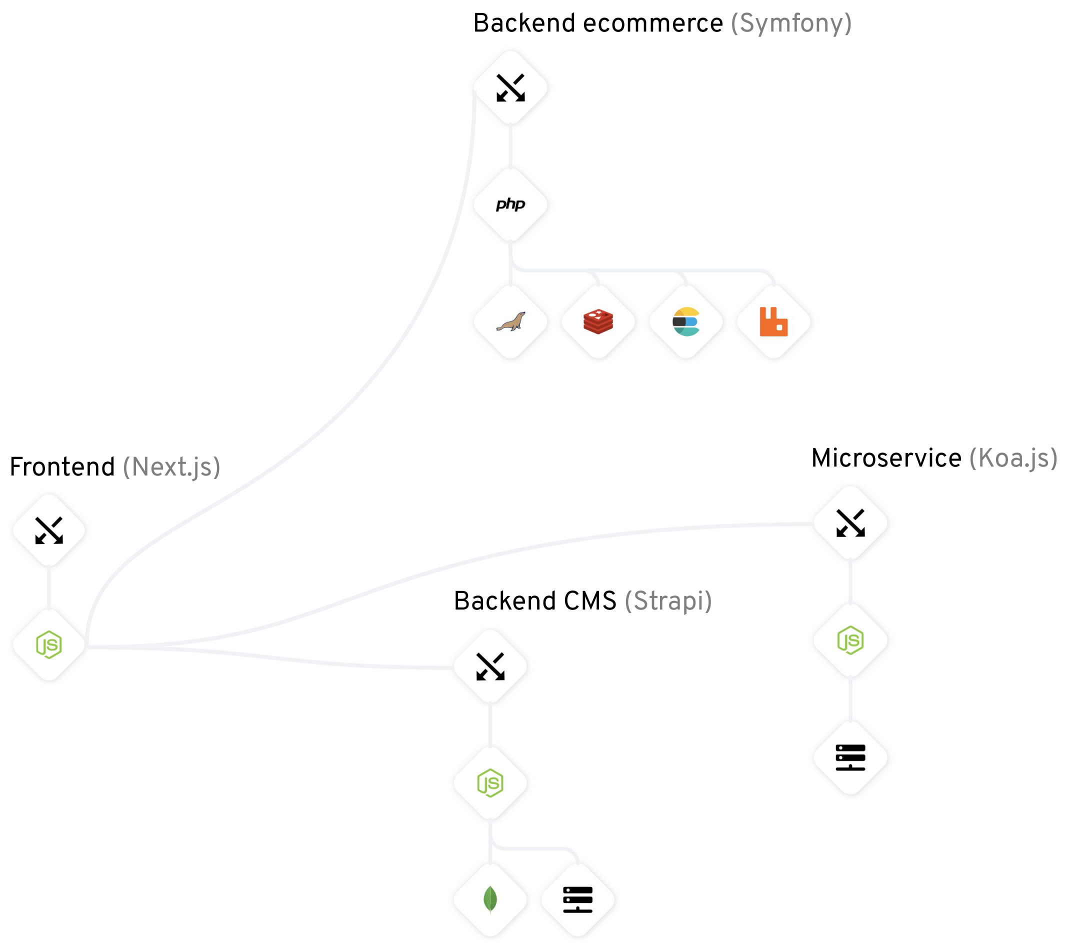


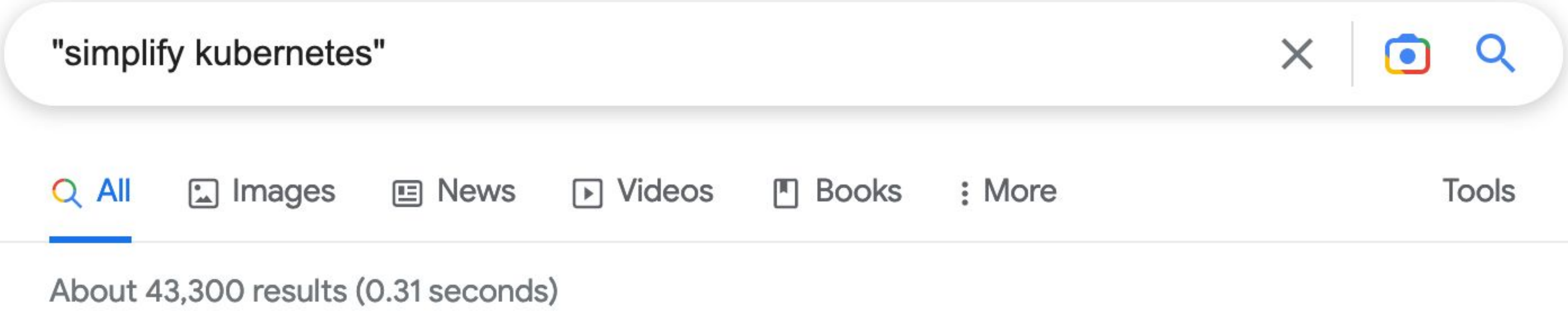
Just an app, but API first + Microservices



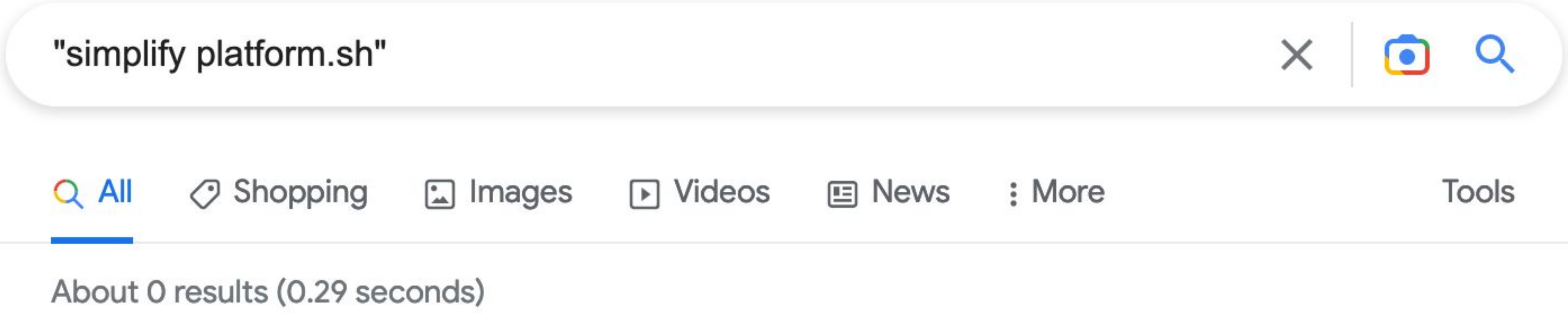
Composable Cloud Infrastructure

Global consumer goods company




"simplify kubernetes"

About 43,300 results (0.31 seconds)


"simplify platform.sh"

About 0 results (0.29 seconds)

build
anything
together

Thank you!

//

Ori Pekelman

Chief Strategy Officer,
Platform.sh

platform.sh